ADAPTIVE THREAD ID CACHE MECHANISM FOR AUTONOMIC PERFORMANCE TUNING

BACKGROUND OF THE INVENTION

[0001] 1. *Field of the Invention*

[0002] The present invention generally relates to computing devices, more specifically, the present invention relates to a processor architecture having an improved caching system.

[0003] 2. *Description of the Related Art*

[0004] Memory access is essential in any computer system and substantially affects the performance of a computer system. Much advancement has been made to improve memory access and among them, use of cache memory to store data that is most likely to be next accessed in fast-coupling memory typically on the main processor.

[0005] Cache memory improves computer's performance when a desired data is found in the cache memory, but the cache memory does not contain all the data needed for a particular application. Where a cache miss occurs, i.e., a needed data is not found in the cache memory, the needed data must be brought in from another slower memory and data from the cache memory must be removed to yield the space for this needed data.

[0006] Cache misses increase especially when a computer is executing in a simultaneous multi-threading mode. In a multi-threading mode, multiple applications access the memory simultaneously and a cache miss by one application may thrash the cache for a second application by removing a data needed by the second application and thus causing a cache miss for the second application.

[0007] As the size of cache memory increases, each cache memory access yields more than one set of data. For example, in a 32 KB cache memory, each access retrieves two pieces of data. After the two pieces of data is retrieved from the cache memory, additional steps must be taken to select one of them for the application's use, thus adding more delay to the data access. This additional

delay becomes especially aggravated when the number of data simultaneously retrieved increases.

SUMMARY OF THE INVENTION

[0008] The invention introduces a way to inhibit data cache thrashing during a multi-threading execution mode through simulating a higher level of associativity in a data cache. An apparatus according to the invention includes at least one instruction register having a thread ID indicator, an address generator having a cache index indicator and a plurality of cache index bits, a cache memory, and a selector for selecting between the thread ID indicator and the cache index indicator. The selector outputs an upper index indicator. When the thread ID indicator is selected by the selector, the thread ID indicator is output to the upper index indicator, and the upper index indicator is concatenated with the plurality of cache index bits to form an address for retrieving an entry from the cache memory.

[0009] In another aspect, the invention is a method for inhibiting data cache thrashing in a multi-threading execution mode through simulating a higher level of associativity in a data cache. The method includes the steps of loading at least one instruction register having a thread ID indicator, generating an effective address having a cache index indicator and a plurality of cache index bits, selecting an upper index indictor between the thread ID indicator and the cache index indicator, forming an address by concatenating the upper index indicator with the plurality of cache index bits, and retrieving an entry from the cache memory indicated by the address.

[0010] Other objects, advantages, and features of the present invention will become apparent after review of the hereinafter set forth in Brief Description of the Drawings, Detailed Description of the Invention, and the Claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] Figs. 1A-B illustrate a prior art architecture for a cache access.

[0012] Fig. 2 illustrates architecture for a cache access according to the invention.

[0013] Fig. 3 illustrates an alternative architecture for a cache access.

DETAILED DESCRIPTION OF THE INVENTION

[0014] In this description, like numerals refer to like elements throughout the several views. The invention temporarily divides a cache to inhibit thrashing and to simulate the performance of higher level associativity. The cache is divided using thread ID bits without requiring extra select time.

[0015] Generally, a one level (L1) cache is unable to implement a high set associativity to support a simultaneous multi-threading execution mode without significant costs in area, power, additional access latency, and significant redesign resource and schedule. Fig. 1A illustrates a problem with a L1 cache. An address generator 105 generates an effective address 106 using information from two registers RA 102 and RB 104. The effective address 106 is fed to a cache memory 107 having two sets of data 108 and 110. The effective address is connected to two comparators 114 and 116 for use in a 2-way late selecting unit 118, which will select which data to output to a cache data bus.

[0016] The late selecting unit can become more complex if more data are needed from a single cache memory access. Fig. 1B illustrates an example, when eight comparators will be needed if eight pieces of data 122 are retrieved from the cache memory, and a 8-way late selecting unit 120 is needed.

[0017] Fig. 2 illustrates an architecture 200 according to the invention. A thread ID indicator 222 is added to an instruction register 220. Although only one instruction register 220 is shown, there can be more than one instruction register when the system is in the multi-thread execution mode. The thread ID indicator can one bit or more bits depending how a cache memory is used during a multi-thread execution mode. This thread ID indicator 222 and bit 0 of an effective address from the address generator 205 are connected to a selector 224. Bit 0 of the effective address, shown as element 208, is also known as the cache index indicator. The rest of bits from the effective address are the cache index bits 210. The selector 224 is controlled by a bit (enable cache split indicator) 234 from a machine state register (MSR) 232 and the selector 224 selectively allows either the thread ID indicator or cache index bit 0 be connected to its output.

This output (upper index indicator) is concatenated with the cache index bits 210 from the effective address to form an index into the data cache 207.

[0018] If the system is in a multi-thread execution mode and the operating system, or hardware, is aware of an application that may cause thrashing, the enable cache split indicator 234 will be set, which in turn directs the selector 224 to connect the thread ID indicator 222 to the selector's output. An application may cause thrashing if it involves technical streaming, i.e., loop operation that involves heavy computation, and this may be indicated by a streaming bit being set by the operating system. The thread ID indicator 222 divides the cache 207 into two halves, upper half 228 and lower half 230. The index formed by the thread ID indicator 222 and the rest of effective address 210 retrieves a set of data from either upper half 228 or lower half 230. The 2-way late selecting unit 218 then selects either a data from cache set 0 or cache set 1 to be output onto the cache data bus.

~~[0019]~~ The enable cache split bit 234 is set by the operating system when the cache 207 is divided to support the multi-thread execution mode. By setting the enable cache split bit 214 and using the thread ID indicator 222, the 2-way late selecting unit 218 can be kept simple and have minimal delay.

[0020] The embodiment shown in Fig. 2 minimizes cache thrashing when a cache miss from a first application causes a data needed by a second application to be discarded. By setting the enable cache split bit, dividing the cache into different regions, and associating these regions with different applications, the thrashing is minimized without incurring additional delays with the late selecting unit.

[0021] Fig. 3 illustrates an alternative embodiment 300 for dynamically splitting the cache. A system may be in a cache thrashing situation if there is a substantial number of cache misses. For a system with two applications running, two cache miss counters 318 and 320 can be set up, one for each application. If application 1 has a cache miss, then counter 320 is incremented. If application 0 has a cache miss, then counter 318 is incremented. The cache miss counters

are compared with a reference counter 322. Each cache miss counter 318, 320 is reset when a new application start is started.

[0022] Instructions for each application is loaded in one instruction register, and for a system that supports two simultaneous applications two instruction registers 302, 304 are used. The two instruction registers 302, 304 are identical. The instruction register 304 has a stream ID indicator 308 to indicate the application is in stream mode, i.e., in a computational loop. The stream ID indicator 308 is set by hardware or the operating system. The instruction register 304 also has a valid bit 308 that is normally set to indicate that instruction in the instruction buffer is valid. The valid bit 308 is unset by hardware if there is a branch condition or a cache miss.

[0023] If either instruction register has the stream ID indicator 308 set and the valid bit 310 unset and either cache miss counter exceeds the reference counter 322 and the enable cache split bit 342 also set, then the 2-way late selecting unit 336 will select the thread ID indicator 334. The thread ID indicator 334 is from an instruction register currently accessing the cache memory.

[0024] Although the invention is described in scenarios supporting one or two threads, the invention can be easily implemented to support more threads without departing from the spirit of the invention.

[0025] In the context of the invention, the method may be implemented, for example, by operating portion(s) of a computing device to execute a sequence of machine-readable instructions. The instructions can reside in various types of signal-bearing or data storage media. The media may comprise, for example, RAM registers, or other memory components of the processor.

[0026] While the invention has been particularly shown and described with reference to a preferred embodiment thereof, it will be understood by those skilled in the art that various changes in form and detail maybe made without departing from the spirit and scope of the present invention as set for the in the following claims. Furthermore, although elements of the invention may be described or claimed in the singular, the plural is contemplated unless limitation to the singular is explicitly stated.